**Search Results** Results **1 - 10** of about **29** for **exception handling compiling** "**side effect**" **branch spe**

IP C/HP-UX Compiler Online Help - HP-UX 11i v1.5: **Compiling** & Running HP C Programs
P C/HP-UX Online Help. NOTE: See the **Compiling** & Running HP C Programs section of the HP C Online Help
·agmas. ... stack unwind-based **exception handling** and asynchronous interrupt **handling** as ... of a **branch** is a
ocs.hp.com/en/1552/summ_opt.html - 107k - Cached - More from this site

:VS log for fptools/ghc/rts/GC.c
Default **branch**: MAIN ... incomprehensible when **compiling** normally - thus ... **exception** is, rather than a flag.
**peculation** ...
/s.haskell.org/cgi-bin/cvsweb.cgi/fptools/ghc/rts/GC.c - More from this site

:VS log for fptools/ghc/rts/Schedule.c
VS log for fptools/ghc/rts/Schedule.c. Keyword substitution: kv. Default **branch**: MAIN. Nuked dead code. Now th
·ocessHeapClosureForDead, pop, and push. ... **exception**. We don't want to have to freeze a bloated stack into t
**peculation** ...
/s.haskell.org/cgi-bin/cvsweb.cgi/fptools/ghc/rts/Schedule.c - 525k - Cached - More from this site

low Control
alling Standard for Alpha Systems: Tru64 UNIX Version 5.1
30097.www3.hp.com/.../DOCUMENTATION/V51_HTML/ARH9MBTE/FLWCNTRL.HTM - 64k - Cached - More fr

ttp://www.helsinki.fi/atk/unix/dec_manuals/DOC_40D/APY8ACTE/DOCU_003.HTM
Flow Control. This chapter contains descriptions of those aspects of the calling standard that deal with the flow o
ıanipulation. That topic is discussed in Chapter 4.
ww.helsinki.fi/atk/unix/dec_manuals/DOC_40D/APY8ACTE/DOCU_003.HTM - 73k - Cached - More from this site

:alling Standard for Alpha Systems
Flow Control. This chapter contains descriptions of those aspects of the calling standard that deal with the flow o
ıanipulation. That topic is discussed in Chapter 4.
ww.iagu.net/docs/dec/AA-PY8AB-TET1_html/callstd4.html - 73k - Cached - More from this site

ttp://tunes.org/~nef/logs/lisp/04.03.13
04:23 <btbngr> Structured **Exception Handling** 07:04:30 <d-bug ... of the techniques for **compiling** a common
nguage, sans ...
nes.org/~nef/logs/lisp/04.03.13 - 183k - Cached - More from this site

ttp://www.cs.arizona.edu/icon/ftp/newsgrp/group95c.txt
it is difficult to check out every **side effect** of what you change. A reminiscence ... files, uncompress using lharc)
ww.cs.arizona.edu/icon/ftp/newsgrp/group95c.txt - 221k - Cached - More from this site

ttp://www.math.utah.edu/pub/tex/bib/toplas.twx
ı%= %% WARNING: Do NOT edit this file.
ww.math.utah.edu/pub/tex/bib/toplas.twx - 135k - Cached - More from this site

ttp://mail.python.org/pipermail/python-announce-list/2002-December.txt
hose who like Python and are familiar with JSP, or PHP, or ASP, should have a look at this engine. It allows one
/namic parts.
ıail.python.org/pipermail/python-announce-list/2002-December.txt - 203k - Cached - More from this site

**Results Page:**
1 2 ▶ **Next**

**Web** | Images | Video | Directory | Local | News | Products

Your Search: exception handling compiling "side effect" branch speculation [Search]

**Web** Results **1** - **10** of about **107** for **exception thrown compiling speculation "side effect" branch**. (**0.17** se

[PDF] Partial Method Compilation using Dynamic Profile Information
File Format: PDF/Adobe Acrobat - View as HTML
... However, **compiling** at a sub-method granularity is thought ... case that an **exception** is **thrown**. They then remove depen- ...
www.eecg.toronto.edu/~tsa/crgpapers/oopsla_final.pdf - Similar pages

[PS] Partial Method Compilationusing Dynamic Profile Information John ...
File Format: Adobe PostScript - View as Text
... However, **compiling** at a sub-method granularity is thought to be too ...
state that needs to be preserved in the case that an **exception** is **thrown**. ...
www.stanford.edu/~jwhaley/papers/oopsla01.ps - Similar pages

[PDF] Intel Technology Journal
File Format: PDF/Adobe Acrobat - View as HTML
... or if it is raised only after some visible **side effect** that. should not have occurred. ... dispatch nodes: a **thrown exception** is represented by an ...
developer.intel.com/technology/itj/2003/ volume07issue01/vol7iss1_managed_runtime_technologies.pdf - Similar pages

UGO.com Forums - The Final Hero
... For easier reference, I'm **compiling** my summaries of the 1stG on these next
... added **side effect** of the botched operation was an increase in body hair, ...
forums.ugo.com/showthread.php?goto=newpost&t=22050 - 171k - Cached - Similar pages

[PDF] Department of Computing Science Compiler Architecture using a ...
File Format: PDF/Adobe Acrobat - View as HTML
... If any other **exception** were **thrown** by f or g, control would flow ... **COMPILING** C–. 79. at the **branch** target. Similarly, the callee-saves registers are ...
www.cs.nott.ac.uk/~fxr/ papers/2002_phd/reig_phd_2002.pdf - Similar pages

languagehat.com: February 2004 Archives
... I will check the Mid-Manhattan **Branch's** catalog for them next time I'm there.
... As a **side effect** of a fruitless search for material on Armeno-Kipchak, ...
www.languagehat.com/archives/2004_02.php - 161k - Cached - Similar pages

[PDF] Techniques for Obtaining High Performance in Java Programs
File Format: PDF/Adobe Acrobat
... is handled by quickly **compiling** a method. into an unoptimized executable code.
The ... Support for **Exception** Handling, Multithreading, and Garbage ...
portal.acm.org/ft_gateway.cfm?id=367714&type=pdf - Similar pages

[PS] Techniques for Obtaining High Performance in Java Programs Iffat H ...
File Format: Adobe PostScript - View as Text
... **exception** handling code at the end of the method, the static **branch** predictor on
... For instance, the calculation of a subscript may, as a **side-effect**, ...
www.arctic.umn.edu/~ihkazi/pub/survey_paper.ps - Similar pages

[PS] c fl Copyright by Scott Alan Mahlke, 1996 EXPLOITING INSTRUCTION ...
File Format: Adobe PostScript - View as Text

... 66 4.4 Example of **exception** detection using sentinel **speculation**. ...
for **speculation** and function calls that do not modify memory (**side-effect** free). ...
www.eecs.umich.edu/~mahlke/ papers/1997/mahlke_phdthesis97.ps - Similar pages

[PDF] Humanist Ethics 'Intelligent' design
File Format: PDF/Adobe Acrobat - View as HTML
... the general election had the **side effect** of completely ... of Constance ordered his bones to be burnt and **thrown** into a ...
www.nzarh.org.nz/journal/summer02.pdf - Similar pages

Gooooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

Free! Get the Google Toolbar. Download Now - About Toolbar

Google ▾ | Search Web ▾ | PageRank | 3 blocked | AutoFill | Options

exception thrown compiling speculat | Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

Welcome United States Patent and Trademark Office

Search Results

BROWSE SEARCH IEEE XPLORE GUIDE

Results for "( exception<in>metadata ) <and> ( speculation<in>metadata )"
Your search matched **5** of **1142142** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

e-mail

» View Session History

» New Search

» **Key**

| | |
|---|---|
| IEEE JNL | IEEE Journal or Magazine |
| IEE JNL | IEE Journal or Magazine |
| IEEE CNF | IEEE Conference Proceeding |
| IEE CNF | IEE Conference Proceeding |
| IEEE STD | IEEE Standard |

**Modify Search**

( exception<in>metadata ) <and> ( speculation<in>metadata )

☐ Check to search only within this results set

**Display Format:** [✓ Citation] Citation & Abstract

**Select** **Article Information**

☐ 1. **Code reordering and speculation support for dynamic optimization systems**
Nystrom, E.M.; Barnes, R.D.; Merten, M.C.; Hwu, W.W.;
Parallel Architectures and Compilation Techniques, 2001. Proceedings. 2001 International Conference on
8-12 Sept. 2001 Page(s):163 - 174
AbstractPlus | Full Text: PDF(1120 KB) **IEEE CNF**

☐ 2. **Eliminating exception constraints of Java programs for IA-64**
Ishizaki, K.; Inagaki, T.; Komatsu, H., Nakatani, T.;
Parallel Architectures and Compilation Techniques, 2002. Proceedings. 2002 International Conference on
22-25 Sept. 2002 Page(s):259 - 268
AbstractPlus | Full Text: PDF(426 KB) **IEEE CNF**

☐ 3. **The Transmeta Code Morphing/spl trade/ Software: using speculation, recovery, and adaptive retransl real-life challenges**
Dehnert, J.C.; Grant, B.K.; Banning, J.P.; Johnson, R.; Kistler, T.; Klaiber, A.; Mattson, J.;
Code Generation and Optimization, 2003. CGO 2003. International Symposium on
23-26 March 2003 Page(s):15 - 24
AbstractPlus | Full Text: PDF(391 KB) **IEEE CNF**

☐ 4. **Integrated predicated and speculative execution in the IMPACT EPIC architecture**
August, D.I.; Connors, D.A.; Mahlke, S.A.; Sias, J.W.; Crozier, K.M.; Ben-Chung Cheng; Eaton, P.R.; Olaniran W.;
Computer Architecture, 1998. Proceedings. The 25th Annual International Symposium on
27 June-1 July 1998 Page(s):227 - 237
AbstractPlus | Full Text: PDF(112 KB) **IEEE CNF**

☐ 5. **A unified compiler framework for control and data speculation**
Dz-Ching Ju, R.; Nomura, K.; Mahadevan, U.; Le-Chun Wu;
Parallel Architectures and Compilation Techniques, 2000. Proceedings. International Conference on
15-19 Oct. 2000 Page(s):157 - 168
AbstractPlus | Full Text: PDF(1016 KB) **IEEE CNF**

View Selected Items

Help Contact Us Privacy

Terms used
**exception speculate branch compiling dag handling**

Found **35** of **153,034**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 35

Result page: **1** 2 next

Relevance scale

1 Parallelizing nonnumerical code with selective scheduling and software pipelining
Soo-Mook Moon, Kemal Ebcioğlu
November 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 6

Full text available: pdf(543.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Instruction-level parallelism (ILP) in nonnumerical code is regarded as scarce and hard to exploit due to its irregularity. In this article, we introduce a new code-scheduling technique for irregular ILP called "selective scheduling" which can be used as a component for superscalar and VLIW compilers. Selective scheduling can compute a wide set of independent operations across all execution paths based on renaming and forward-substitution and can compute availab ...

**Keywords**: VLIW, global instruction scheduling, instruction-level parallelism, software pipelining, speculative code motion, superscalar

2 Efficient superscalar performance through boosting
Michael D. Smith, Mark Horowitz, Monica S. Lam
September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems**, Volume 27 Issue 9

Full text available: pdf(1.63 MB) Additional Information: full citation, abstract, references, citings, index terms

The foremost goal of superscalar processor design is to increase performance through the exploitation of instruction-level parallelism (ILP). Previous studies have shown that speculative execution is required for high instruction per cycle (IPC) rates in non-numerical applications. The general trend has been toward supporting speculative execution in complicated, dynamically-scheduled processors. Performance, though, is more than just a high IPC rate; it also depends upon instruction count ...

3 Evaluation of scheduling techniques on a SPARC-based VLIW testbed
Seongbae Park, SangMin Shim, Soo-Mook Moon
December 1997 **Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture**

Full text available:

pdf(1.40 MB) Publisher Site   Additional Information: full citation, abstract, references, citings, index terms

The performance of Very Long Instruction Word (VLIW) microprocessors depends on the close cooperation between the compiler and the architecture. This paper evaluates a set of important compilation techniques and related architectural features for VLIW machines. The evaluation is performed on a SPARC-based VLIW testbed where gcc-generated optimized SPARC code is scheduled into high-performance VLIW code. As a base scheduling compiler, we experiment with three core scheduling techniques including ...

**Keywords**: SPARC-based VLIW testbed, VLIW microprocessors, Very Long Instruction Word microprocessors, all-path speculation, compiler, computer architecture, copies, gcc-generated optimized SPARC code, high-performance VLIW code, loop unrolling, memory disambiguation, nongreedy enhanced pipeline scheduling, nonspeculative operations, parallel machines, performance, profile-based all-path speculation, renaming, restricted speculative loads, scheduling compiler, scheduling techniques, software pipelining, speculative operations, trace-based speculation

4 An efficient resource-constrained global scheduling technique for superscalar and VLIW processors
Soo-Mook Moon, Kemal Ebcioğlu
December 1992 **ACM SIGMICRO Newsletter , Proceedings of the 25th annual international symposium on Microarchitecture**, Volume 23 Issue 1-2
Full text available: pdf(2.05 MB)   Additional Information: full citation, references, citings, index terms

**Keywords**: VLIW, compile-time parallelization, instruction-level parallelism, superscalar

5 A region-based compilation technique for a Java just-in-time compiler
Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani
May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation**, Volume 38 Issue 5
Full text available: pdf(158.62 KB)   Additional Information: full citation, abstract, references, citings, index terms

Method inlining and data flow analysis are two major optimization components for effective program transformations, however they often suffer from the existence of rarely or never executed code contained in the target method. One major problem lies in the assumption that the compilation unit is partitioned at method boundaries. This paper describes the design and implementation of a region-based compilation technique in our dynamic compilation system, in which the compiled regions are selected a ...

**Keywords**: dynamic compilers, on-stack replacement, partial inlining, region-based compilation

6 Effectiveness of cross-platform optimizations for a java just-in-time compiler
Kazuaki Ishizaki, Mikio Takeuchi, Kiyokuni Kawachiya, Toshio Suganuma, Osamu Gohda, Tatsushi Inagaki, Akira Koseki, Kazunori Ogata, Motohiro Kawahito, Toshiaki Yasue, Takeshi Ogasawara, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Additional Information:

Full text available: pdf(405.65 KB) full citation, abstract, references, citings, index terms

This paper describes the system overview of our Java Just-In-Time (JIT) compiler, which is the basis for the latest production version of IBM Java JIT compiler that supports a diversity of processor architectures including both 32-bit and 64-bit modes, CISC, RISC, and VLIW architectures. In particular, we focus on the design and evaluation of the cross-platform optimizations that are common across different architectures. We studied the effectiveness of each optimization by selectively disabling ...

**Keywords**: Java, just-in-time compiler, optimization

7 Post-pass binary adaptation for software-based speculative precomputation
Steve S.W. Liao, Perry H. Wang, Hong Wang, Gerolf Hoflehner, Daniel Lavery, John P. Shen
May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5
Full text available: pdf(330.60 KB) Additional Information: full citation, abstract, references, citings, index terms

Recently, a number of thread-based prefetching techniques have been proposed. These techniques aim at improving the latency of single-threaded applications by leveraging multithreading resources to perform memory prefetching via speculative prefetch threads. Software-based speculative precomputation (SSP) is one such technique, proposed for multithreaded Itanium models. SSP does not require expensive hardware support-instead it relies on the compiler to adapt binaries to perform prefetching on o ...

**Keywords**: chaining speculative precomputation, delay minimization, dependence reduction, long-range thread-based prefetching, loop rotation, pointer, post-pass, prediction, scheduling, slack, slicing, speculation, triggering

8 A study on the number of memory ports in multiple instruction issue machines
Soo-Mook Moon, Kemal Ebcioğlu
December 1993 **Proceedings of the 26th annual international symposium on Microarchitecture**
Full text available: pdf(1.28 MB) Additional Information: full citation, references, citings

**Keywords**: ILP, memory disambiguation, memory ports, speculative loads, static scheduling

9 A study of source-level compiler algorithms for automatic construction of pre-execution code
Dongkeun Kim, Donald Yeung
August 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 3
Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, index terms

Pre-execution is a promising latency tolerance technique that uses one or more helper threads running in spare hardware contexts ahead of the main computation to trigger long-latency memory operations early, hence absorbing their latency on behalf of the main computation. This article investigates several source-to-source C compilers for extracting pre-execution thread code automatically, thus relieving the programmer or hardware from this onerous task. We present an aggressive profile-driven co ...

**Keywords**: Data prefetching, memory-level parallelism, multithreading, pre-execution, prefetch conversion, program slicing, speculative loop parallelization

10 From checking to inference via driving and dag grammars
Jens Peter Secher, Morten Heine Sørensen
January 2002 **ACM SIGPLAN Notices , Proceedings of the 2002 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation**, Volume 37 Issue 3
Full text available: pdf(365.48 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Abramov and Glück have recently introduced a technique called URA for inverting first order functional programs. Given some desired output value, URA computes a potentially infinite sequence of substitutions/restrictions corresponding to the relevant input values. In some cases this process does not terminate.In the present paper, we propose a new program analysis for inverting programs. The technique works by computing a finite grammar describing the set of all input that relate to a given ...

**Keywords**: inference, program inversion, supercompilation

11 Split-path enhanced pipeline scheduling for loops with control flows
SangMin Shim, Soo-Mook Moon
November 1998 **Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.52 MB) Additional Information: full citation, references, citings, index terms

**Keywords**: all-path pipelining, enhanced pipeline scheduling, initiation interval, modulo scheduling, multi-path loops, software pipelining

12 Memory forwarding: enabling aggressive layout optimizations by guaranteeing the safety of data relocation
Chi-Keung Luk, Todd C. Mowry
May 1999 **ACM SIGARCH Computer Architecture News , Proceedings of the 26th annual international symposium on Computer architecture**, Volume 27 Issue 2
Full text available: pdf(196.77 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

By optimizing data layout at run-time, we can potentially enhance the performance of caches by actively creating spatial locality, facilitating prefetching, and avoiding cache conflicts and false sharing. Unfortunately, it is extremely difficult to guarantee that such optimizations are *safe* in practice on today's machines, since accurately updating *all* pointers to an object requires perfect alias information, which is well beyond the scope of the compiler for languages such as C. T ...

13 A provably time-efficient parallel implementation of full speculation
John Greiner, Guy E. Blelloch
January 1996 **Proceedings of the 23rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(1.45 MB) Additional Information: full citation, references, citings, index terms

14 Interprocedural conditional branch elimination
Rastislav Bodík, Rajiv Gupta, Mary Lou Soffa
May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation**, Volume 32 Issue 5
Full text available: pdf(2.02 MB) Additional Information: full citation, abstract, references, citings, index terms

The existence of statically detectable correlation among conditional branches enables their elimination, an optimization that has a number of benefits. This paper presents techniques to determine whether an interprocedural execution path leading to a conditional branch exists along which the branch outcome is known at compile time, and then to eliminate the branch along this path through code restructuring. The technique consists of a demand driven interprocedural analysis that determines whethe ... .

15 Instruction scheduling for the Motorola 88110
Mark Smotherman, Shuchi Chawla, Stan Cox, Brian Malloy
December 1993 **Proceedings of the 26th annual international symposium on Microarchitecture**
Full text available: pdf(702.49 KB) Additional Information: full citation, references, citings

**Keywords:** MC88110, cache alignment, instruction scheduling, superscalar processors

16 Whole program paths
James R. Larus
May 1999 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation**, Volume 34 Issue 5
Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, references, citings, index terms

*Whole program paths (WPP)* are a new approach to capturing and representing a program's dynamic---actually executed---control flow. Unlike other path profiling techniques, which record intraprocedural or acyclic paths, WPPs produce a single, compact description of a program's entire control flow, including loop iteration and interprocedural paths.This paper explains how to collect and represent WPPs. It also shows how to use WPPs to find *hot subpaths*, which are the heavily executed ...

**Keywords**: data compression, dynamic program measurement, path profiling, program control flow, program tracing

17 Inducing heuristics to decide whether to schedule
John Cavazos, J. Eliot, B. Moss
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6
Full text available: pdf(352.40 KB) Additional Information: full citation, abstract, references, index terms

Instruction scheduling is a compiler optimization that can improve program speed, sometimes by 10% or more, but it can also be expensive. Furthermore, time spent optimizing is more important in a Java just-in-time (JIT) compiler than in a traditional one because a JIT compiles code at run time, adding to the running time of the program. We found that, on any given block of code, instruction scheduling often does not produce significant benefit and sometimes degrades speed. Thus, we hoped that we ...

**Keywords**: Java, Jikes RVM, compiler optimization, instruction scheduling, machine learning, supervised learning

18 Improving balanced scheduling with compiler optimizations that increase instruction-level parallelism
Jack L. Lo, Susan J. Eggers
June 1995 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1995 conference on Programming language design and implementation**, Volume 30 Issue 6
Full text available: pdf(1.35 MB) Additional Information: full citation, abstract, references, citings, index terms

Traditional list schedulers order instructions based on an optimistic estimate of the load latency imposed by the hardware and therefore cannot respond to variations in memory latency caused by cache hits and misses on non-blocking architectures. In contrast, balanced scheduling schedules instructions based on an estimate of the amount of instruction-level parallelism in the program. By scheduling independent instructions behind loads based on what the program can provide, rather than what ...

19 Phase coupling for horizontal microcode generation
Vicki H. Allan, Robert A. Mueller
December 1987 **Proceedings of the 20th annual workshop on Microprogramming**
Full text available: pdf(1.16 MB) Additional Information: full citation, abstract, references, citings

Microcode for mass produced architectures is still predominantly generated by hand. Yet, as speed dictates the vertical migration of commonly executed functions to microcode, the demand for automated code generation increases. Though considerably more complex than phase-decoupled methods, phase-coupled methods for the generation of horizontal microcode have the potential to produce more highly optimized microcode. Results of the retargetable phase-coupled microcode compiler, Hori ...

20 Space-efficient scheduling of nested parallelism
Girija J. Narlikar, Guy E. Blelloch
January 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 1
Full text available: pdf(481.02 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Many of today's high-level parallel languages support dynamic, fine-grained parallelism. These languages allow the user to expose all the parallelism in the program, which is typically of a much higher degree than the number of processors. Hence an efficient scheduling algorithm is required to assign computations to processors at runtime. Besides having low overheads and good load balancing, it is important for the scheduling algorithm to minimize the space usage of the parallel program. T ...

**Keywords**: dynamic scheduling, multithreading, nested parallelism, parallel language implementation, space efficiency

Results 1 - 20 of 35 Result page: **1** 2 next

CiteSeer Find: exception AND handling AND specul Documents Citations

Searching for **exception and handling and speculation**.
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Google (CiteSeer) Google (Web) Yahoo! MSN CSB DBLP
24 documents found. **Order: number of citations.**

A Scalable Approach to Thread-Level Speculation - Steffan, Colohan, Zhai, Mowry (2000) (Correct) (30 citations)
was not designed to scale to larger systems. One **exception** (prior to this publication) is a proposal by with matrix-based programs, and its success in **handling** pointer-based codes has yet to be demonstrated. A Scalable Approach to Thread-Level **Speculation** J. Gregory Steffan, Christopher B. Colohan,
www.cs.cmu.edu/~steffan/items/isca00.ps

RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory.. - Vijay Pai (1997) (Correct) (27 citations)

File Integer Register File Completion Graduation **Exception Handling** Register mapping (renaming) Active Register File Completion Graduation **Exception Handling** Register mapping (renaming) Active List ALU/ list, the number of shadow mappers for branch **speculation**, and the size of the branch prediction
www-ece.rice.edu/~sarita/Publications/tcca1097.ps

Specifying superscalar microprocessors in Hawk - Cook, Launchbury, Matthews (1998) (Correct) (14 citations)
record the original instruction sequence for **exception handling**. ffl A processor that renames the original instruction sequence for **exception handling**. ffl A processor that renames registers must based on the Intel P6 with features such as **speculation**, register renaming, and superscalar
www.cse.ogi.edu/PacSoft/projects/Hawk/papers/super.ps

A Scalable Register File Architecture for Dynamically.. - Wallace, Bagherzadeh (1996) (Correct) (11 citations)
register requirements for high performance and **exception handling** can be quite high. Farkas et. al. file needs 2N read ports and N write ports to **handle** the worst-case scenario. The area complexity for the number of branch checkpoints. With increased **speculation** from wideissue superscalars and larger mapping
www.eng.uci.edu/students/swallace/papers_wallace/pact96.ps

Unconstrained Speculative Execution with Predicated State Buffering - al. (1995) (Correct) (8 citations)
the side effects and appropriately **handles exceptions** caused by the speculative execution.
www.shimada.nuee.nagoya-u.ac.jp/~ando/pub/ISCA95.ps.gz

Code Reordering and Speculation Support for Dynamic.. - Nystrom, Barnes.. (2001) (Correct) (6 citations)
basic block boundaries, both the ordering of **exceptions** and the observed processor register contents at
www.crhc.uiuc.edu/IMPACT/ftp/conference/pact-01-speculation.ps

Sentinel Scheduling with Recovery Blocks - August, Deitrich, Mahlke (1995) (Correct) (4 citations)
of the application. This includes ignoring **exceptions** during the extra executions of potentially **speculation**'s solution to proper **exception handling** is to provide a non-excepting version of each of coping with high memory latency. As such, **speculation** can also diminish the negative effects of
www.crhc.uiuc.edu/IMPACT/ftp/report/crhc-95-05.sentinel.ps.Z

A Static Study of Java Exceptions using JESP - Ryder, Smith, Kremer, Gordon.. (1999) (Correct) (2 citations)
A Static Study of Java **Exceptions** using JESP Barbara G. Ryder, Donald Smith,
www.cs.rutgers.edu/~uli/CC00.ps

Cherry: Checkpointed Early Resource Recycling in.. - Martinez, Renau.. (2002) (Correct) (1 citation)
on state checkpointing and rollback to service **exceptions** for instructions whose resources have been instruction or its successors. Then, the **exception handler** is invoked. One disadvantage of typical ROB load/store queue entries in processors with load **speculation** and replay traps. Overall, this Cherry
www.csl.cornell.edu/~martinez/doc/micro02.ps

Delayed Exceptions - Speculative Execution of Trapping.. - Ertl, Krall (1994) (Correct) (1 citation)
Springer LNCS 786, 1994, pages 158-171 Delayed **Exceptions** -Speculative Execution of Trapping

davinci.snu.ac.kr/links/ilp/ertl94.ps.gz

Java Compilation for Multi-threaded Architectures - Arvind Hossell Koppe (2001) (Correct)
a conditional branch, a switched branch, an **exception** throw, or a return from the entire method.
thanks to its stack orientation and its **handling** of constants, such as the class or field
parallelism, and data and control **speculation** techniques to improve performance. The paper
www.icsa.informatics.ed.ac.uk/cpc2001/Proceedings/rangaswami.ps

Speculative Software Management of Datapath-width.. - Pokam, Rochecouste, .. (2004) (Correct)
and the register widths to the compiler. Simple **exception** management allows this exposition to be only
a simple and e#cient recovery mechanism for **handling** such width mispredictions. The remainder of this
In this sense, we introduce datapath-width **speculation** principally as a means to predict, at the
www.irisa.fr/caps/people/seznec/specwidth.pdf

Realizing High IPC Through a Scalable Memory-Latency.. - Multipath.. (Correct)
components of the microarchitecture. With the **exception** of the execution window block, this is similar
decoded after being fetched. All further **handling** of the instructions is done in their decoded
since our design utilizes a form of value **speculation** on all operands, we consume the present value
www.ece.neu.edu/info/architecture/publications/medea02.pdf

Analysis of the Effectiveness of Multithreading for.. - Pattery, Lee, Won (2002) (Correct)
as follows. Related work in the field of **exceptions** and embedded processors is discussed in the
context, cache misses due to fetching interrupt **handle**r from memory, and the possibility of NIC buffer
either by the compiler or dynamically through **speculation** [13]and scheduled onto specially modified
www.ece.orst.edu/%7Ebenl/Publications/ccn2002.pdf

The StarJIT Compiler: A Dynamic Compiler for Managed Runtime.. - Adl-Tabatabai (2003) (Correct)
phase establishes basic block boundaries and **exception handling** regions, and it recovers type
establishes basic block boundaries and **exception handling** regions, and it recovers type information for
(such as global scheduling and control **speculation**) for performance. A dynamic compiler must
developer.intel.com/technology/itj/2003/volume07issue01/art02_starjit/vol7iss1_art02.pdf

Phi-Predication for Light-Weight If-Conversion - Chuang, Calder, Ferrante (2003) (Correct)
aggressive **speculation**. Any memory updates and **exception handling** are qualified by the predicate. A
**speculation**. Any memory updates and **exception handling** are qualified by the predicate. A qualified
of load and store instructions for aggressive **speculation**. Any memory updates and **exception handling** are
www-cse.ucsd.edu/~calder/papers/CGO-03-PHI.pdf

IA-64 Floating-Point Operations and the IEEE Standard for.. - Cornea-Hasegan, Norin (Correct)
are also illustrated. IA-64 floating-point **exceptions** and traps are described, including the Software
with the IEEE Standard's recommendations in **handling** floating-point **exceptions** are specified. The
developer.intel.com/technology/itj/q41999/pdf/ia64fpbf.pdf

A Dynamically Adaptive Parallelization Model Based on Speculative.. - Kazi (2000) (Correct)
Features .73 3.5.1 **Exception Handling** .
. 73 3.5.1 **Exception Handling** .
run-time data-dependence checking and control **speculation** to parallelize such loops. Next, a
www.arctic.umn.edu/papers/kazi-phd-thesis-00.ps.gz

*First 20 documents* Next 20

Try your query at: Google (CiteSeer) Google (Web) Yahoo! MSN CSB DBLP